

Základní pojmy

Jan Faigl

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Přednáška 1

A0B36PR1 – Programování 1

Část 1 – Organizace předmětu

Informace o předmětu

Přednášky

Cvičení a domácí úkoly

Hodnocení předmětu a zkouška

Část 2 – Programování a výpočty

Výpočty a výpočetní prostředky

Program a programovací jazyk

Část 3 – Programovací jazyk Java

Programovací jazyk Java

Příklad jednoduchého programu

Základní datové typy

Část I

Organizace předmětu

Organizace a hodnocení předmětu

- A0B36PR1 – Programování 1

- Rozsah: 2p+2c

- Zakončení: Z,ZK

Z – zápočet, ZK – zkouška

- Kredity: 6

V prvním semestru je nutné získat ≥ 15 kreditů

Základní zdroje a webové stránky

A0B36PR1 - Programování 1

- Webové stránky předmětu

<https://edux.feld.cvut.cz/courses/A0B36PR1>

- Odevzdávání domácích úkolů

<https://cw.felk.cvut.cz/upload>

- Přednášející:

- Ing. **Jan Faigl**, Ph.D.



- doc. Ing. **Ivan Jelínek**, CSc.



Cíle předmětu

Programování 1

Prerekvizita Programování 2 a Algoritmizace

- **Osvojit si** pohled na výpočetní prostředky jako „*počítačový vědec*“ a naučit se je efektivně používat *Computer scientist*

- Formulovat problém a jeho řešení počítačovým programem
 - Získat povědomí jaké problémy lze výpočetně řešit

- **Získat zkušenost** s programováním

získání vlastní zkušenosti

- Programování v jazyku Java

cvičení a domácí úkoly

- **Osvojit si** schopnost číst, psát a porozumět malých programům


- **Získat** programovací návyky jak psát

- srozumitelné a přehledné zdrojové kódy;
 - opakovaně použitelné programy.

Zdroje a literatura

- Přednášky – slidy, poznámky a především **vlastní zápisky**
- Cvičení – získání praktických dovedností řešením domácích úkolů a dalších úloh

programovat, programovat, programovat

 Učebnice jazyka Java 5. v., *Pavel Herout* KOPP, 2010, ISBN 978-80-7232-398-2



 Introduction to Java Programming, 9th Edition, *Y. Daniel Liang* Prentice Hall, 2012



<http://www.cs.armstrong.edu/liang/intro9e>

- On-line kurzy programování (v Java nebo jiném jazyku)

- <http://www.algoritmy.net/article/21340/Uvod-1>
- http://www.linuxsoft.cz/article.php?id_article=244
- <http://www.root.cz/serialy/programovaci-jazyk-java-a-jvm>

g programování java

Cvičení

- Ing. **Martin Balík**, Ph.D.



- Ing. **Jakub Mrva**



- Ing. **Zdeněk Buk**, Ph.D.



- Ing. **Jiří Šebek**



- Ing. **Jan Faigl**, Ph.D.



- Ing. **Martin Schaefer**



- Ing. **Ondřej Hrstka**



- RNDr. **Jan Tožička**



Přednášky – zimní semestr (ZS) akademického roku 2014/2015

- Harmonogram akademického roku 2014/2015

<http://www.fel.cvut.cz/cz/education/harmonogram1415.html>

- Přednášky:

- **Otevřená informatika (OI)**: středa, 16:15–17:45

Dejvice, místnost T2:D3-309

- **Kybernetika a robotika (KyR)**: čtvrtek, 15:15–17:00

Karlovo náměstí, místnost KN:E-107

- 13 výukových týdnů

- Rektorské volno – středa 1. 10. od 16:00

Výuka na FEL probíhá

Počítačové laboratoře

- Sítové domovské adresáře

NFS v4

Přenos a synchronizace souborů

- USB media
- Sítové přenosy (ftp, ssh, unison, rsync)
- Owncloud – <https://owncloud.cesnet.cz>

služby sdružení CESNET

- Programování ve vývojovém prostředí NetBeans IDE 8.0 a Java verze 8.

<https://netbeans.org>

- Použití libovolného jiného prostředí je možné, např. IntelliJ IDEA, Eclipse,

<https://download.cvut.cz>

- či kombinace ant a editoru vim.

- Odevzdávání domácích úkolů

- Upload System – <https://cw.felk.cvut.cz/upload>

Domácí úkoly a další úlohy

- Samostatná práce s cílem osvojit si praktické zkušenosti
- Odevzdání domácích úkolů prostřednictvím Upload System
 - <https://cw.felk.cvut.cz/upload>
 - Nahrátí (upload) archivů s nezbytnými zdrojovými soubory
 - Ověření správnosti implementace automatickými testy
 - detekce plagiátů*
- Úkoly jsou jednoduché a navrhované tak, aby byly stihnutelné
- Klíčem k úspěšnému dokončení předmětu je samostatná práce a osvojení si technik a znalostí
 - průběžná práce a řešení úkolů*
- Pokud něčemu nerozumíte, ptejte se cvičících
 - pokud možno hned a neodkládejte na později*
- Pokud vám přijde úkolů málo, ptejte se po dalších úlohách na procvičování.

Kontrola znalostí testy

- 4 testy na cvičení každý se ziskem maximálně 5 bodů
 1. (týden 5) – psací test (~ 15-20 minut)
 - „programování“ na papír*
 2. (týden 8) – implementace programu (~ 60 minut)
 - odevzdání prostřednictvím Upload System*
 3. (týden 10) – implementace programu (~ 60 minut)
 - odevzdání prostřednictvím Upload System*
 4. (týden 12) – psací test (~ 60 minut)
 - „programování“ na papír*
- Uvedené časy jsou orientační a spíše odpovídají očekávané náročnosti testu*

Přehled domácích úkolů

- 10 domácích úkolů po 5 bodech (+1 testovací)
 0. (týden 1) - Robot Karel *testovací úkol za 0 bodů*
 1. (týden 2) - Robot Karel s cykly
 2. (týden 3) - Robot Karel s vlastním příkazem
 3. (týden 4) - Jednoduchá kalkulačka
 4. (týden 5) - Zpracování vstupu a výstupu programu
 5. (týden 6) - Výpočet průměrné hodnoty a směrodatné odchylky
 6. (týden 7) - Automatické zpracování souboru hodnot
 7. (týden 8) - Implementace kruhové fronty
 8. (týden 9) - Řešení problému Hanojských věží rekurzí
 9. (týden 11) - Zatřídování spojových seznamů
 10. (týden 12) - Implementace prioritní fronty haldou
- Podmínkou zápočtu je úspěšné odevzdání všech domácích úkolů
- Bodová ztráta za pozdní odevzdání úkolu
 - Maximální počet bodů za úkol klesá s každým týdnem pozdního odevzdání*

Hodnocení předmětu

Zdroj bodů	Maximum bodů	Přípustné minimum bodů
Domácí úkoly (10×5 bodů)	50	30
Testy na cvičení (4×5 bodů)	20	10
Písemný zkouškový test	20	10
Ústní zkouška	20	-10

- Minimální počet bodů pro zápočet **40**
- Při rozhodnutí k ústní zkoušce je odečteno 10 bodů
- Pro úspěšné absolvování předmětu je nutné získat **zápočet** a vykonat **zkoušku**
- Získání **zápočtu** je podmíněno odevzdáním všech domácích úkolů a úspěšně složenými testy
 - Podmínka nutná nikoliv však postačující*

Klasifikace předmětu

Klasifikace	Bodové rozmezí	Hodnocení	Slovní hodnocení
A	90–100	1	výborně
B	80–89	1,5	velmi dobře
C	70–79	2	dobře
D	60–69	2,5	uspokojivě
E	50–59	3	dostatečně
F	<50	4	nedostatečně

- Minimální přípustné body:
30 (úkoly) + 10 (testy) + 10 (písemná zkouška) = 50 bodů

Základní koncept programování

„Separating Programming Sheep from Non-Programming Goats”

<http://blog.codinghorror.com/separating-programming-sheep-from-non-programming-goats>
<http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf>

- Efektivní metody výuky programování se hledají již od dob prvních počítačů
tj. přes více než 50 let
- Přesto se zdá, že je každý základní kurz programování obtížný a 30% až 60% studentů jej na poprvé nezvládne
- Základní koncept je pochopení principu přiřazení hodnoty proměnné

Část II

Programování a výpočetní prostředky

Test pochopení principu přiřazení

- Zápis programu pro přiřazení hodnot do proměnných *a* a *b* a následné přiřazení proměnné *b* do *a*.

Přiřazení hodnoty proměnné

```
1 int a = 10;
2 int b = 20;
3
4 a = b;
```

- Jaké jsou hodnoty proměnných *a* a *b*?

- | | |
|--------------------------------|--------------------------------|
| a. <i>a</i> = 20 <i>b</i> = 0 | f. <i>a</i> = 30 <i>b</i> = 0 |
| b. <i>a</i> = 20 <i>b</i> = 20 | g. <i>a</i> = 10 <i>b</i> = 30 |
| c. <i>a</i> = 0 <i>b</i> = 10 | h. <i>a</i> = 0 <i>b</i> = 30 |
| d. <i>a</i> = 10 <i>b</i> = 10 | i. <i>a</i> = 10 <i>b</i> = 20 |
| e. <i>a</i> = 30 <i>b</i> = 20 | j. <i>a</i> = 20 <i>b</i> = 10 |

Skupiny počítačových uživatelů

„Uživatel“

- Spouštěč programů
- Zadává vstup
- Čeká na výstup
- Čte výstup

Píše, kliká

- **Relativně omezená množina vstupů**

Pouze to co je dovoleno

„Programátor“

- Spouští programy
- Dává počítači příkazy
- Vytváří nové programy
- Kombinuje příkazy

Řadí je do posloupnosti

- **Rozmanitější možnosti použití**

Omezen pouze limity počítače

Výpočetní prostředky (počítače)

- Jednoúčelové přístroje s předepsaným chováním
 - program / posloupnost kroků (instrukcí) je vestavěná a neměnná

Kalkulačka, pračka, první telefony
- Počítač s uloženým programem v paměti
 - Posloupnost instrukcí čtena z paměti
 - Flexibilita ve tvorbě posloupnosti

Program lze libovolně měnit
- Architektura počítače se společnou pamětí pro data a program
 - Von Neumannova architektura počítače

John Louis von Neumann (1903–1957)

Způsob reprezentace znalostí

- Z hlediska výpočtu můžeme rozlišit dva základní typy znalostí:

Způsob popisu problému

Deklarativní

- Tvrzení popisující stav
- Axiomatické
- Umožňuje jednoduše ověřovat (testovat) pravdivost tvrzení
- Neposkytuje návod jak vypočíst hodnotu

Příklad:

$$\sqrt{x} = y, y^2 = x, x \geq 0, y \geq 0$$

Imperativní

- Popis jak něco vypočítat
- Posloupnost výpočtu
- Test jak ovlivnit průběh výpočtu

Příklad:

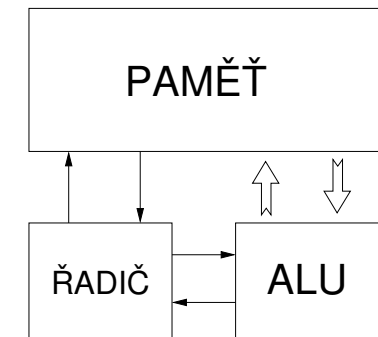
1. If $y^2 \approx x$
2. Then
 - return y**
3. Else
 - $y \leftarrow \frac{y+x}{2}$
 - Go to Step 2

Von Neumannova architektura

- ALU - Aritmeticko logická jednotka (Arithmetic Logic Unit)

Základní matematické a logické instrukce
- PC - Čítač instrukcí (Program Counter)

„Ukazuje“ na místo v paměti s instrukcemi pro vykonání



V drtivě většině případů je program posloupnost instrukcí zpracovávající jednu nebo dvě hodnoty (uložené v nějakém paměťovém místě) jako vstup a generování nějaké výstupní hodnoty, kterou ukládá někam do paměti nebo modifikuje hodnotu PC (podmíněné řízení běhu programu).

Program je „recept“

- Program je posloupnost kroků (výpočtů) popisující průběh výpočtu pro řešení problému
(je to „recept“ na řešení problému)
- Pro zápis receptu potřebujeme **jazyk**
Způsob zápisu programu
- Jazyk definuje základní sadu primitiv (operací/příkazů), které můžeme použít pro zápis receptu
- S konečnou množinou primitiv dobrý programátor naprogramuje „cokoliv“.
Co může být vyjádřeno.
- Turing Machine – obecný model počítačícího stroje
Alan Turing, 1936
- V předmětu A0B36PR1 používáme programovací jazyk **Java**
Programování není o znalosti konkrétního programovacího jazyka, je to o způsobu uvažování a řešení problému.

Definice programovacího jazyka

- **Syntax** – definice povolených výrazů a konstrukcí programu
Plná kontrola a podpora vývojových prostředí
 - Příklad popisu výrazu gramatikou v **Backus-Naurově formě**.
 $\langle \text{exp} \rangle ::= \langle \text{exp} \rangle + \langle \text{exp} \rangle \mid \langle \text{exp} \rangle * \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \mid \langle \text{number} \rangle \langle \text{number} \rangle ::= \langle \text{number} \rangle \langle \text{digit} \rangle \mid \langle \text{digit} \rangle$
 $\langle \text{digit} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
- **Statická sémantika** – definuje jak jsou konstrukty používány
Částečná kontrola a podpora prostředí
 - Příklad axiomatické specifikace: $\{P\} S \{Q\}$, P - precondition, Q -postcondition, S - konstrukce jazyka.
- **Plná sémantika** – co program znamená a dělá, jeho smysluplnost
Kontrola a ověření správnosti je kompletně v režii programátora.

Programovací jazyk

- Existuje množství programovacích jazyků
- Nelze říci, že jeden jazyk je lepší než druhý
V podstatě jsou všechny ekvivalentní
- Můžeme, ale říci, že některé jazyky se hodí na konkrétní úlohy
- Základní dělení:
 - Vysoko-úrovňové a nízko-úrovňové
Liší se mohutností množiny primitiv
 - Obecné a speciální (určené pro konkrétní aplikace)
 - Interpretované a překládané
 - Dle typu: imperativní (procedurální), funkcionální, logické (deklarativní), objektově-orientované

Správnost programu

- Syntakticky i staticky sémanticky správný program neznamena, že dělá to co od něj požadujeme
- Správnost a smysluplnost programu je dána očekávaným chování při řešení požadovaného problému
- V zásadě při spuštění programu mohou nastat tyto události:
 - Program havaruje a dojde k chybovému výpisu
Mrzuté, ale výpis (report) je dobrý start řešení chyby (bug)
 - Program běží, ale nezastaví se a počítá v nekonečné smyčce.
Zpravidla velmi obtížné detekovat a program ukončujeme po nějaké době.
 - Program včas dává odpověď
Je však dobré vědět, že odpověď je korektní.

Správnost programu je plně v režii programátora, proto je důležité pro snadnější ověření správnosti, ladění a hledání chyby používat **dobrý programovací styl.**

Program a jeho zápis

- Program – popis činnosti prováděné počítačem.
- Programovací jazyk – notační systém pro zápis programu.
Program zpravidla zapisujeme ve zdrojových (textových) souborech.
- Čitelnost programu:
 - strojová – efektivnost kódu
 - lidská – srozumitelnost, udržovatelnost, **kódovací konvence**,
Vývojová prostředí (editor, debugger, nástroje pro správu verzí, analýza, testování → softwarové inženýrství.
- Abstrakce
 - Datová – základní typy, struktury, modulární.
 - Řídící – základní, strukturální, modulární.

Java

- Obecný, vyšší, imperativní (procedurální) a objektově orientovaný jazyk
- Překládaný jazyk zaměřený na přenositelnost (portabilitu) zdrojových kódů i přeložených binárních souborů
- Historie:
 - 1991 – nejdříve jako jazyk Oak
 - 1995 – Java JDK 1 (první veřejná verze)
 - 1998 – Java 2 (ver. 1.2)
 - 2002 – Java 2 (ver. 1.4) a J2EE
 - 2004 – Java 2 (ver. 1.5) – J2SE5.0
 - 2011 – Java 7 (vydává Oracle po akvizici Sun Microsystems)
 - 2014 – Java 8 (18. března, 2014)
- Součástí základního vývojového prostředí je bohatý soubor knihovných funkcí.

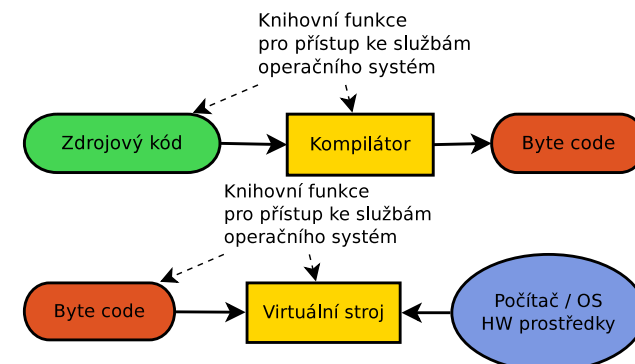
Java je relativně jednoduchý jazyk (v základní verzi) a jeho efektivní používání je spíše o znalosti knihovných funkcí.

Část III

Úvod do programovacího jazyku Java

Zdrojové kódy, překlad a spuštění Java programu

- Zdrojové kódy jsou zapisovány v textových souborech s koncovkou **.java**
- Zdrojové soubory jsou překládány překladačem (javac) do binárního kódu („byte code“) uložených v souborech s koncovkou **.class**
- Spuštění programu je realizováno virtuálním strojem, který poskytuje abstrakci na operačním systému počítače



Java prostředí – JDK, JRE, JVM

- **JDK** (Java Development Kit) – základní vývojové prostředí, knihovny funkcí, překladač zdrojových souborů javac. Jeho součástí je i JRE.
- **JRE** (Java Runtime Environment) – základ prostředí Java pro spouštění programů, obsahuje virtuální stroj java.
- **JVM** (Java Virtual Machine) – virtuální stroj pro spouštění Java programů (java).
- **JAR** (Java ARchive) – archiv Java souborů, typicky množiny zkompileovaných .class (tříd) doplněných textovým popisem (Manifest), kterou třídu spustit. Slouží pro snadnější spouštění programů o více souborech.

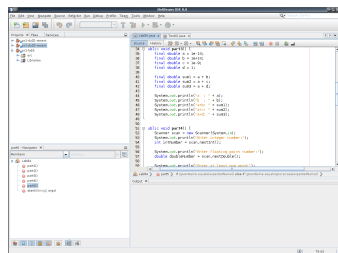
V podstatě ZIP archiv

Integrovaná vývojová prostředí (IDE)

- Nadstavba základních příkazů javac a java
- Integrují (mimo jiné) systém pro řízení překladu
Např. ant nebo maven
- Zvýrazňují syntax, doplňují jména a provádějí základní kontrolu kódu
- Mezi nejznámější patří Netbeans, Eclipse a IntelliJ IDEA

<https://download.cvut.cz>

- Na cvičení je používáno prostředí Netbeans
- Import kódů do projektu
štábní kultura
- Adresářová struktura projektu



Příklad

Výpočet druhé odmocniny

```

1 double x = 13.0;
2 double y = 1.0;
3 int i = 1;
4
5 while(Math.abs(y*y - x) > 1e-3) {
6     System.out.println("Step " + i + " y = " + y);
7     y = (y+(x/y))/2;
8     i += 1;
9 }
10 System.out.println("sqrt(" + x + ") found in " + i + "
    steps as " + y);

```

Kompilace a spuštění programu

```

javac sqrt.java
java sqrt

```

Primitivní typy

- Vyhrazuji v paměti místo pro uložení číselné hodnoty
Velikost alokovaného prostoru odpovídá možnému rozsahu a přesnosti reprezentace čísla
- Celočíselné typy reprezentují čísla v definovaném rozsahu
- Neceločíselné typy reprezentují reálná, racionální a iracionální čísla.

Aproximace do pevného počtu bitů

- Mantisa + exponent

$s \ e \ e \ \dots \ M \ M \ \dots \ M$, kde s - znaménkový bit,

M - mantisa, e - exponent,

$s \times M \times B^{e-E}$, B - báze, E - konstanta.

- double - 64 bitů podle IEEE 754, $\pm 4.9E-324 - \pm 1.7E+308$.

Možnosti zvýšení přesnosti

- Reprezentace racionálních čísel - podíl dvou celočíselných hodnot, např. *Homogenní souřadnice*.
- „Libovolná přesnost“ - speciální knihovny, např. *gmp* až do výše volné paměti.

souřadnice x,y - 7511164176768 346868669952 3739567104 ~ 2008.57, 92.76

Zápis číselné hodnoty v programu (literál)

- Základní zápis je pro typ **int** a **double**

```
int decI = 173;
int hexI = 0xad;
int binI = 0b10101101;
```

```
double d1 = 105.67;
double d2 = 1.0567e2;
```

- Zápis hodnoty typu **long** a **float** je nutné specifikovat

```
long l1 = 171; //specifikujeme znakem l
long l2 = 13313514L; //nebo znakem L
```

```
float f1 = 105.67f; //nutne uvest f nebo F
float f2 = 1.0567e2f; //jinak pri kompilaci chyba
```

Základní typy v Javě

Celočíselné typy

- **byte** – 8 bitů (1 byte),
-128 až 127
- **short** – 16 bitů (2 byte),
-32 768 až 32 767
- **int** – 32 bitů (4 bytes),
-2³¹ až 2³¹-1
základní celočíselný typ
- **long** – 64-bitů (8 bajtů),
-2⁶³ až 2⁶³-1

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

Neceločíselné typy

- **float** – 32-bit IEEE 754
- **double** – 64-bit IEEE 754

Logický a znakový typ

- **boolean** – true / false
- **char** – jeden 16-bit Unicode znak

Přesnost výpočtu 1/2

- Ztráta přesnosti při aritmetických operacích.

Příklad sčítání dvou čísel

```
1 public static void main(String[] args) {
2     double a = 1E+10;
3     double b = 1E-10;
4     System.out.println("a : " + a);
5     System.out.println("b : " + b);
6     System.out.println("a+b: " + (a+b));
7 }
8
9 javac Sum.javac
10 java Sum
11 a : 1.0E10
12 b : 1.0E-10
13 a+b: 1.0E10
```

Přesnost výpočtu 2/2

Příklad dělení dvou čísel

```

1 public static void main(String[] args) {
2     final int number = 100;
3     double dV = 0.0;
4     float fV = (float)0.0;
5     for (int i = 0; i < number; i++) {
6         dV += 1.0 / 10.0;
7         fV += 1.0 / 10.0;
8     }
9     System.out.println("double value: " + dV);
10    System.out.println("float value: " + fV);
11 }
12
13 javac Division.java
14 java Division
15 double value: 9.999999999999998
16 float value: 10.000002

```

Zdroje a typy chyby

- Chyby matematického modelu - matematická aproximace fyzikální situace.
- Chyby vstupních dat.
- Chyby numerické metody.
- Chyby zaokrouhlovací.
- Absolutní chyba aproximace
 $E(x) = \hat{x} - x$, \hat{x} přesná hodnota, x aproximace.
- Relativní chyba $RE(x) = \frac{\hat{x}-x}{x}$.

Přesnost výpočtu - strojová přesnost

- Strojová přesnost ϵ_m - nejmenší desetinné číslo, které přičtením k 1.0 dává výsledek různý od 1, pro $|v| < \epsilon_m$, platí

$$v + 1.0 == 1.0.$$

- Zaokrouhlovací chyba - nejméně ϵ_m .
- Přesnost výpočtu - aditivní chyba roste s počtem operací v řádu $\sqrt{N} \cdot \epsilon_m$.
 - Často se však kumuluje preferabilně v jedno směru v řádu $N \cdot \epsilon_m$.

Podmíněnost numerických úloh

- Podmíněnost úlohy $C_p = \frac{\text{relativní chyba výstupních údajů}}{\text{relativní chyba vstupních údajů}}$
- Dobře podmíněná úloha $C_p \approx 1$.
- Výpočet je dobře podmíněný, je-li málo citlivý na poruchy ve vstupních datech.
- Numericky stabilní výpočet - vliv zaokrouhlovacích chyb na výsledek je malý.
- Výpočet je stabilní, je-li dobře podmíněný a numericky stabilní.

Příklady chyb

- Ariane 5 - 4.6.1996
40 sekund po startu explodovala. Datová konverze z 64-bitového desetinné reprezentace na 16-ti bitový znaménkový integer.
http://www.esa.int/esaCP/Pr_33_1996_p_EN.html
- Systém Patriot - 25.2.1991
Systémový čas v desetinách sekundy, převod na sekundy realizován dělením 10, registry pouze 24 bitů.
<http://www.ima.umn.edu/~arnold/disasters/patriot.html>
<http://www5.informatik.tu-muenchen.de/~huckle/bugse.html>

Shrnutí přednášky

Diskutovaná témata

- Inforamce o předmětu
- Programování, reprezentace znalostí, program a programovací jazyk
- Úvod do jazyku Java
- Základní datové typy a přesnost výpočtu

- **Příště: Výrazy, operátory a řídicí struktury**